

An Outline of OpenAI's $2^{o(n)}$ -Time k -SAT Algorithm

October 2026

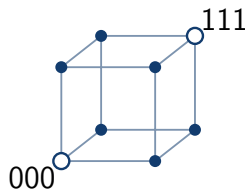
k -SAT: decision and counting

A k -SAT formula is an AND of m clauses, each an OR of at most k literals (Booleans $x_1, \dots, x_n$ or negations $\neg x_1, \dots, \neg x_n$).

Example for $k = 3$:

$$F = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3).$$

The number of satisfying assignments is $\#F = 8 - 2 = 6$.



Complexity of deciding whether $\#F > 0$:

- Linear $O(n + m)$ time for $k = 2$ (Aspvall–Plass–Tarjan, 1979);
- NP-complete for every fixed $k \geq 3$ (Cook, 1971; Karp, 1972).

Searching for a satisfying assignment

3-SAT on n bits has 2^n possible assignments. Two notable randomized search algorithms improving on brute force:

- WalkSAT: randomly flip a bit in a violated clause. Finds a solution in $O^*((4/3)^n)$ time on average (Schöning, 1999).
- PPSZ: assign bits in random order (Paturi–Pudlák–Saks–Zane, 1998). Use a forced value when a few clauses imply it; otherwise guess. $O(1.308^n)$ (Hertli, 2011).

SAT and fine-grained complexity

Exponential Time Hypothesis (ETH); (Impagliazzo–Paturi, 2001)

For some $c > 0$, 3-SAT admits no deterministic $O(2^{cn})$ -time algorithm.

Strong Exponential Time Hypothesis (SETH)

For every $0 < \varepsilon < 1$, there is a fixed k for which no deterministic algorithm solves k -SAT in time $O(2^{(1-\varepsilon)n})$.

Reductions from SAT gave conditional lower bounds for familiar problems.

- Euclidean traveling salesman. (de Berg–Bodlaender–Kisfaludi-Bak–Kolay, 2018).
- Edit distance. (Bringmann–Künnemann, 2015).
- Nearest neighbor search. (Rubinstein, 2018).

The Exponential Time Hypothesis is false

Theorem (OpenAI/Anthropic, 2026)

Fix $k \geq 2$. Given k -SAT on n variables and m clauses, a deterministic algorithm counts its solutions in polynomial space and subexponential time:

$$(n + m)^{O(1)} 2^{O(\rho(n))} \text{ bit operations,} \quad \rho(n) = \frac{n \log \log n}{\sqrt{\log n}} = o(n).$$

Even weaker tasks had been conjectured to require exponential time:

- Random ETH: refuting sparse random 3-SAT (Song–Woodruff–Zhong, 2017).
- NSETH: checking a proof that a k -SAT is unsatisfiable (Carmosino–Gao–Impagliazzo–Mihajlin–Paturi–Schneider, 2016).

We'll focus on getting $2^{o(n)}$, skipping some quantitative optimizations.

Other consequences: Max-SAT and graph colorings

Max- k -SAT: Let $S(\mathbf{x})$ count the clauses satisfied by $\mathbf{x}$. For fixed $k \geq 2$ and $\varepsilon > 0$, subsample $O(n/\varepsilon^2)$ random clauses.

- Can encode “ $\mathbf{x}$ solves at least t clauses” as SAT problem of size $O_\varepsilon(n)$.
- In $2^{o(n)}$ time, find $\mathbf{x}$ satisfying the most subsampled clauses (encode “counting satisfied clauses” with addition circuits). Then whp, it nearly maximizes the fraction of original clauses satisfied, up to error ε .

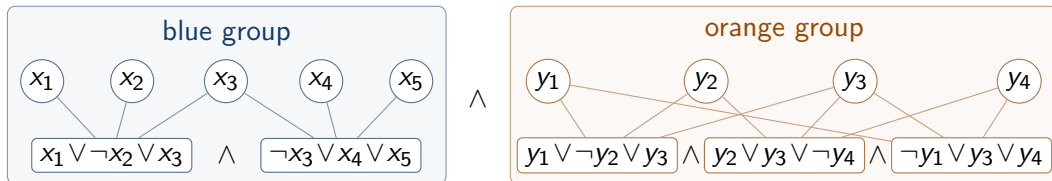
Graph colorings: For fixed $q \geq 2$, can count the proper q -colorings of any simple v -vertex graph deterministically in $2^{o(v)}$ time.

E.g. for three colors, exactly one of $\text{blue}_u, \text{orange}_u, \text{purple}_u$ is true at each vertex u . On an edge uv , $\neg \text{blue}_u \vee \neg \text{blue}_v$ forbids two blue endpoints.

Proof overview for 3-SAT

The algorithm is guided by **wishful thinking**. In the idealized case:

- 1 The variables can be partitioned into r color classes, each with at most $b \leq o(n)$ variables.
- 2 Every clause is **monochromatic**, using variables of only 1 color.



Then solutions can be counted within each color class separately, in time $O^*(2^b)$:

$$\#F = \prod_{\text{colors } c} \#F_c.$$

Proof overview for 3-SAT

The algorithm is guided by **wishful thinking**. In the idealized case:

- ① The variables can be partitioned into r color classes, each with at most $b \leq o(n)$ variables.
- ② Every clause is **monochromatic**, using variables of only 1 color.

This ideal case will not actually happen, but the proof makes it “almost” happen:

- I. Reduce to the case of near-linearly many clauses via sparsification.
- II. Algebrize, defining a univariate low-degree polynomial with $H(0) = \#F$.
- III. $H(1), \dots, H(Q)$ will behave like monochromatic CSPs (here $Q \rightarrow \infty$ slowly).
- IV. Compute $H(1), \dots, H(Q)$ and higher derivatives $H^{(j)}(t)$.
- V. Recover $H(0)$ by polynomial interpolation on the data from part IV.
 - We'll need $\deg H \leq o(Qn)$ so $o(n)$ -order derivatives suffice for interpolation.

Part I. Reducing to sparse formulas

Counting sparsification (Impagliazzo–Paturi–Zane, 2001)

Given n -variable 3-SAT F with m clauses and an integer $s \geq 2$, construct 3-SAT problems $F_1, \dots, F_L$ on the same variables in $\text{poly}(n, m, s) 2^{n/s}$ time. Their satisfying sets partition that of F .

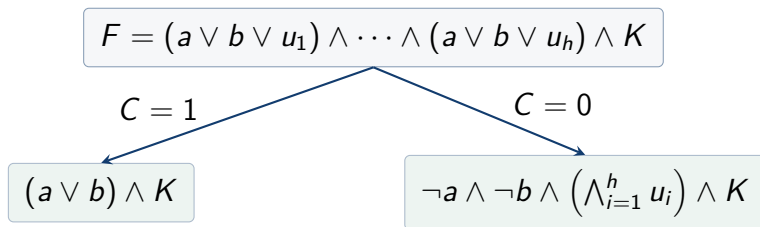
$$\#F = \sum_{i=1}^L \#F_i, \quad L \leq \text{poly}(n, s) 2^{n/s}.$$

In each F_i , every variable occurs at most $O((s \log s)^2)$ times.

We'll take $s \rightarrow \infty$ slowly with n .

Example step of sparsification

The standard sparsification algorithm uses repeated “branching”. For example: suppose $C = a \vee b$ appears many times. Write K for all other clauses in F .



The two branches partition the satisfying assignments, and are simpler:

- The $C = 1$ case has removed several clauses.
- The $C = 0$ case has “fixed” all the variables u_i , effectively removing them.

Part II. Algebrizing the SAT formula

A Boolean formula such as $F = (x \vee y) \wedge (\neg x \vee \neg y)$ can be Algebrized:

$$x \vee y \quad \rightarrow \quad \Phi_1(x, y) = 1 - (1 - x)(1 - y),$$

$$\neg x \vee \neg y \quad \rightarrow \quad \Phi_2(x, y) = 1 - xy.$$

Part II. Algebrizing the SAT formula

A Boolean formula such as $F = (x \vee y) \wedge (\neg x \vee \neg y)$ can be Algebrized:

$$\begin{aligned}x \vee y &\rightarrow \Phi_1(x, y) = 1 - (1 - x)(1 - y), \\ \neg x \vee \neg y &\rightarrow \Phi_2(x, y) = 1 - xy.\end{aligned}$$

The product $\Phi_1\Phi_2$ is the indicator of F . Summing over (x, y) counts solutions:

$$\#F = \sum_{(x,y) \in \{0,1\}^2} \Phi_1(x, y) \Phi_2(x, y) = 2.$$

In general,

$$\#F = \sum_{\sigma \in \{0,1\}^n} \prod_{e \in F} \Phi_e(\sigma).$$

Extension by T

For each variable v in clause e , we'll choose a linear form $\ell_{e,v}(T)$ in the bits, with polynomial dependence on T and $\ell_{e,v}(0) = v$. E.g.:

Clause i	x -input $\ell_{i,x}(T)$	y -input $\ell_{i,y}(T)$
1 $(x \vee y)$	$(1 + T)x + T^2y$	$y + 3Tx$
2 $(\neg x \vee \neg y)$	$x - Ty$	$(1 + T^2)y + 2T^3x$

Extension by T

For each variable v in clause e , we'll choose a linear form $\ell_{e,v}(T)$ in the bits, with polynomial dependence on T and $\ell_{e,v}(0) = v$. E.g.:

Clause i	x -input $\ell_{i,x}(T)$	y -input $\ell_{i,y}(T)$
1 $(x \vee y)$	$(1 + T)x + T^2y$	$y + 3Tx$
2 $(\neg x \vee \neg y)$	$x - Ty$	$(1 + T^2)y + 2T^3x$

The polynomial $H(T)$ is defined by summing over inputs as before:

$$H(T) = \sum_{(x,y) \in \{0,1\}^2} \Phi_1(\ell_{1,x}(T), \ell_{1,y}(T)) \Phi_2(\ell_{2,x}(T), \ell_{2,y}(T)).$$

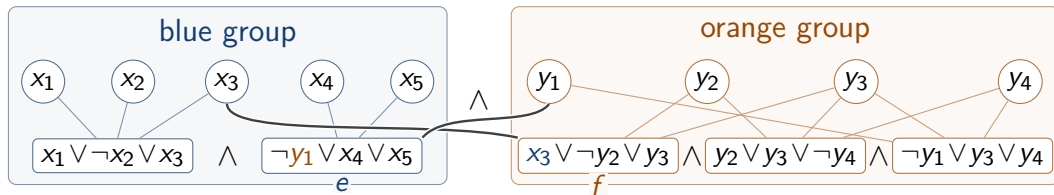
At $T = 0$, both clauses receive (x, y) , so $H(0) = \#F$.

We'll algebraize so each clause has low T -degree. Sparsification then means $H(T)$ is low-ish degree, so we won't need to compute too many derivatives at nonzero t .

Part III. Computing $H(t)$ one color at a time

Start with i.i.d. uniform colors $a_{v,t}$ for variables and $d_{e,t}$ for clauses at $t = 1, \dots, Q$. Colors live in $[r]$ for $r \asymp \sqrt{\log n}/(\log \log n)^2$.

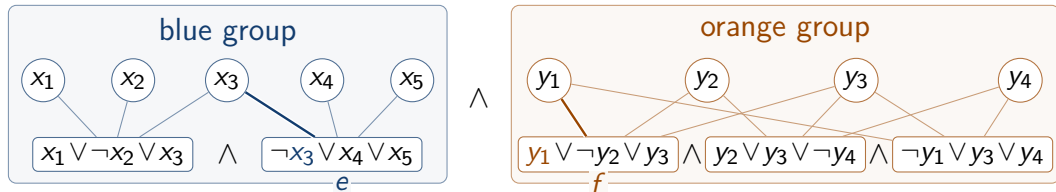
Variables and clauses at $t = 1$



Part III. Computing $H(t)$ one color at a time

Start with i.i.d. uniform colors $a_{v,t}$ for variables and $d_{e,t}$ for clauses at $t = 1, \dots, Q$. Colors live in $[r]$ for $r \asymp \sqrt{\log n}/(\log \log n)^2$.

Example after substituting the two crossed inputs at $T = 1$



$$\ell_{e,y_1}(T) = (1 - T)y_1 + Tx_3,$$

$$\ell_{f,x_3}(T) = (1 - T)x_3 + Ty_1.$$

Color lists and the full transpose

For each assignment $\mathbf{x} \in \{0, 1\}^n$, form its table of clause inputs $\mathbf{Z}(\mathbf{x})$. For occurrence (v, e) , set $Z_{a_v, d_e}(\mathbf{x}) = x_v$. Empty cells store 0.

$a_v \backslash d_e$	BB	BO	OB	OO
BB	$(v_1, e_1)(v_2, e_3)(v_1, e_5)(v_2, e_7)$			
BO	$(v_3, e_1)(v_4, e_3)(v_3, e_5)$	$\emptyset$		
OB	$(v_5, e_1)(v_5, e_3)(v_6, e_5)(v_6, e_7)$			
OO	$(v_7, e_2)(v_7, e_4)(v_8, e_6)(v_8, e_7)$			

occurrences (v, e)

Color lists and the full transpose

For each assignment $\mathbf{x} \in \{0, 1\}^n$, form its table of clause inputs $\mathbf{Z}(\mathbf{x})$. For occurrence (v, e) , set $Z_{\mathbf{a}_v, \mathbf{d}_e}(\mathbf{x}) = x_v$. Empty cells store 0.

$\mathbf{a}_v \backslash \mathbf{d}_e$	BB	BO	OB	OO
BB	$(v_1, e_1)(v_2, e_3)(v_1, e_5)(v_2, e_7)$			
BO	$(v_3, e_1)(v_4, e_3)(v_3, e_5)$	$\emptyset$		
OB	$(v_5, e_1)(v_5, e_3)(v_6, e_5)(v_6, e_7)$			
OO	$(v_7, e_2)(v_7, e_4)(v_8, e_6)(v_8, e_7)$			

occurrences (v, e)

store x_v →

$\mathbf{a}_v \backslash \mathbf{d}_e$	BB	BO	OB	OO
BB	x_1	x_2	x_1	x_2
BO	x_3	x_4	x_3	0
OB	x_5	x_5	x_6	x_6
OO	x_7	x_7	x_8	x_8

original inputs $\mathbf{Z}(\mathbf{x})$

Repeated pairs $(\mathbf{a}_v, \mathbf{d}_e) = (\mathbf{a}_{v'}, \mathbf{d}_{e'})$ go in separate tables, each of size $r^Q \times r^Q$.

For simplicity, consider a single table (in general, “do the same thing” per table).

Color lists and the full transpose

For each assignment $\mathbf{x} \in \{0, 1\}^n$, form its table of clause inputs $\mathbf{Z}(\mathbf{x})$. For occurrence (v, e) , set $Z_{\mathbf{a}_v, \mathbf{d}_e}(\mathbf{x}) = x_v$. Empty cells store 0.

$\mathbf{a}_v \backslash \mathbf{d}_e$	BB	BO	OB	OO
BB	$(v_1, e_1)(v_2, e_3)(v_1, e_5)(v_2, e_7)$			
BO	$(v_3, e_1)(v_4, e_3)(v_3, e_5)$	$\emptyset$		
OB	$(v_5, e_1)(v_5, e_3)(v_6, e_5)(v_6, e_7)$			
OO	$(v_7, e_2)(v_7, e_4)(v_8, e_6)(v_8, e_7)$			

occurrences (v, e)

transpose (V) $\rightarrow$

$\mathbf{a}_v \backslash \mathbf{d}_e$	BB	BO	OB	OO
BB	x_1	x_3	x_5	x_7
BO	x_2	x_4	x_5	x_7
OB	x_1	x_3	x_6	x_8
OO	x_2	0	x_6	x_8

swapped inputs $V\mathbf{Z}(\mathbf{x})$

$$V : \mathbb{R}^{r^{2Q}} \rightarrow \mathbb{R}^{r^{2Q}}, \quad [V\mathbf{Z}(\mathbf{x})]_{\mathbf{a}_v, \mathbf{d}_e} = Z_{\mathbf{d}_e, \mathbf{a}_v}(\mathbf{x}).$$

Swapping implements the “variable exchange” in the last slide, for all $1 \leq T \leq Q$.

Want $H(T)$ to implement swapping for $T \in \{1, 2, \dots, Q\}$ (but not for $T = 0$).

Structure of the linear forms

We'll construct a special matrix K based on V and use (with notational abuse) linear forms:

$$\vec{\ell}(\mathbf{x}) = (I - K)\mathbf{Z}(\mathbf{x}) + K\mathbf{Z}(\mathbf{x}) \odot (1 + \langle T \rangle).$$

- 1 At $T = 0$, reduces to the identity $\vec{\ell}(\mathbf{x}) = \mathbf{x}$.
- 2 We'll ensure $\text{rank}(I - K) \leq o(n)$. This means $(I - K)\mathbf{Z}(\mathbf{x})$ is determined by $o(n)$ linear measurements $\mathbf{m}(\mathbf{x})$.
- 3 Computation of $K\mathbf{Z}(\mathbf{x})$ factors into color groups, after fixing $\mathbf{m}(\mathbf{x})$ and fixing $T \in \{1, \dots, Q\}$.
- 4 The polynomials in T will have degree $\ll Q$.

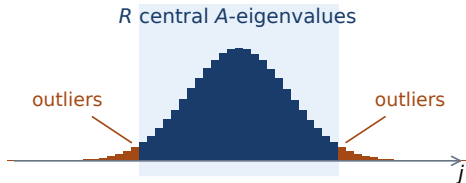
The spectrum of commuting swap operators

Let Π_s exchange the row and column colors at position $1 \leq s \leq Q$. These commute.

$$V = \prod_{s=1}^Q \Pi_s, \quad A = \sum_{s=1}^Q \Pi_s.$$

On a common eigenvector ξ , let $j_\xi = \#\{s : \Pi_s \xi = -\xi\}$. Then

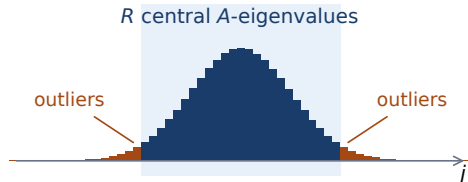
$$A\xi = (Q - 2j_\xi)\xi, \quad V\xi = (-1)^{j_\xi}\xi.$$



The spectrum of commuting swap operators

Let Π_s exchange the row and column colors at position $1 \leq s \leq Q$. These commute.

$$V = \prod_{s=1}^Q \Pi_s, \quad A = \sum_{s=1}^Q \Pi_s.$$



On a common eigenvector ξ , let $j_\xi = \#\{s : \Pi_s \xi = -\xi\}$. Then

$$A\xi = (Q - 2j_\xi)\xi, \quad V\xi = (-1)^{j_\xi}\xi.$$

Pick R with $\sqrt{Q} \ll R \ll Q$. Let f be a polynomial of degree $< R$ with

$$f(Q - 2j_\xi) = (-1)^{j_\xi}$$

on the R central eigenvalues.

- $K = Vf(A)$ fixes the central eigenspaces $\implies \text{rank}(I - K) \leq o(n)$.
- Being able to take $R \ll Q$ is the “crucial gain” in the proof.

Low degree limits the color mismatches

Key Observation (to convert small $\deg f$ to small $\deg \ell_{e,v}$ on next slide)

If row $\mathbf{b}$ differs from the clause's color list $\mathbf{d}_e$ in at least R positions, then the coefficient of $Z_{\mathbf{b},\mathbf{c}}$ in $[K\mathbf{Z}]_{\mathbf{a}_v,\mathbf{d}_e} = [Vf(A)\mathbf{Z}]_{\mathbf{a}_v,\mathbf{d}_e}$ is zero for every column $\mathbf{c}$.

Proof.

Let $\mathcal{M} = \text{span}\{I, \Pi_1, \dots, \Pi_Q\}$. Since $\deg f < R$,

$$f(A) \in \mathcal{M}^{R-1} = \text{span}\left\{ \prod_{s \in S} \Pi_s : S \subseteq [Q], |S| < R \right\}.$$

The transpose V exchanges all Q positions. Each term in $f(A)$ undoes at most $R - 1$ of those swaps in $K = Vf(A)$. □

The choice of low- T -degree linear forms

Our final choice of linear form is a T -deformation of $(I - K)\mathbf{Z} + K\mathbf{Z}$:

$$\ell_{e,v}(T; \mathbf{x}) = \underbrace{[(I - K)\mathbf{Z}(\mathbf{x})]_{\mathbf{a}_v, \mathbf{d}_e}}_{\text{determined by } \mathbf{m}(\mathbf{x})} + \sum_{\mathbf{b}, \mathbf{c} \in [r]^Q} K[(\mathbf{a}_v, \mathbf{d}_e), (\mathbf{b}, \mathbf{c})] Z_{\mathbf{b}, \mathbf{c}}(\mathbf{x}) \prod_{\substack{1 \leq s \leq Q \\ b_s \neq d_{e,s}}} \left(1 - \frac{T}{s}\right).$$

$< R$ terms

- At $T = 0$, all $\left(1 - \frac{T}{s}\right)$ factors are 1, so $\ell_{e,v}(0; \mathbf{x}) = Z_{\mathbf{a}_v, \mathbf{d}_e}(\mathbf{x}) = x_v$.

The choice of low- T -degree linear forms

Our final choice of linear form is a T -deformation of $(I - K)\mathbf{Z} + K\mathbf{Z}$:

$$\ell_{e,v}(T; \mathbf{x}) = \underbrace{[(I - K)\mathbf{Z}(\mathbf{x})]_{\mathbf{a}_v, \mathbf{d}_e}}_{\text{determined by } \mathbf{m}(\mathbf{x})} + \sum_{\mathbf{b}, \mathbf{c} \in [r]^Q} K[(\mathbf{a}_v, \mathbf{d}_e), (\mathbf{b}, \mathbf{c})] Z_{\mathbf{b}, \mathbf{c}}(\mathbf{x}) \prod_{\substack{1 \leq s \leq Q \\ b_s \neq d_{e,s}}} \left(1 - \frac{T}{s}\right). \\ \underbrace{\hspace{10em}}_{< R \text{ terms}}$$

- At $T = 0$, all $\left(1 - \frac{T}{s}\right)$ factors are 1, so $\ell_{e,v}(0; \mathbf{x}) = Z_{\mathbf{a}_v, \mathbf{d}_e}(\mathbf{x}) = x_v$.
- First term: choose an orthogonal basis $\mathcal{E}$ of outlier eigenspaces with entries $\{-1, 0, 1\}$. Measure the $o(n)$ small integers $\mathbf{m}(\mathbf{x}) = (\langle \psi, \mathbf{Z}(\mathbf{x}) \rangle)_{\psi \in \mathcal{E}}$.
- Second term: only terms with $b_T = d_{e,T}$ survive, from $\mathbf{x}|_{B_{T, d_{e,T}}}$, where $B_{T, g} = \{w : a_{w,T} = g\}$ is the set of variables with T -color g .

Conclusion: we can evaluate $\ell_{e,v}(t; \mathbf{x})$ from just $\mathbf{m}(\mathbf{x})$ and $\mathbf{x}|_{B_{T, d_{e,t}}}$.

Final Computation of $H(t)$ from the color classes

Fix t . Each color- g clause Φ_e can be evaluated from $\mathbf{x}_g = \mathbf{x}|_{B_{t,g}}$ and $\mathbf{m}(\mathbf{x})$.

$$H(t) = \sum_{\mathbf{x} \in \{0,1\}^n} \prod_{g=1}^r \underbrace{\prod_{e: d_{e,t}=g} \Phi_e((\ell_{e,v}(t; \mathbf{x}))_{v \in e})}_{P_g(\mathbf{x}_g, \mathbf{m}(\mathbf{x}))}.$$

We decompose the linear measurements $\mathbf{m}(\mathbf{x})$ by the color of variables x_i :

$$\mathbf{m}(\mathbf{x}) = \sum_{g=1}^r \mathbf{m}_g(\mathbf{x}_g).$$

Fix the partial sums $\mathbf{m}_g(\mathbf{x}_g) = \boldsymbol{\mu}_g$ for all g and let $\boldsymbol{\mu} = \sum_g \boldsymbol{\mu}_g$. Then finally:

$$H(t) = \sum_{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_r} \prod_{g=1}^r \left[\sum_{\mathbf{x}_g \in \{0,1\}^{B_{t,g}}; \mathbf{m}_g(\mathbf{x}_g) = \boldsymbol{\mu}_g} P_g(\mathbf{x}_g, \boldsymbol{\mu}) \right].$$

$2^{o(n)}$ choices of $(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_r)$. At most $2^{2n/r}$ assignments for each inner color- g sum.

Part IV. Recovering the count

For each $t = 1, \dots, Q$, we also must compute $H(t), H'(t), \dots, H^{(J)}(t)$, where

$$J = \left\lfloor \frac{3mR}{Q} \right\rfloor, \quad \deg H \leq 3mR < Q(J+1).$$

These $Q(J+1)$ numbers determine $H \in \mathbb{Q}[T]$, and hence $H(0) = \#F$.

Part IV. Recovering the count

For each $t = 1, \dots, Q$, we also must compute $H(t), H'(t), \dots, H^{(J)}(t)$, where

$$J = \left\lfloor \frac{3mR}{Q} \right\rfloor, \quad \deg H \leq 3mR < Q(J+1).$$

These $Q(J+1)$ numbers determine $H \in \mathbb{Q}[T]$, and hence $H(0) = \#F$.

Example for $H'(t)$: Differentiate one input $\ell_{e,v}$ and write $W_{e,v}(\mathbf{x})$ for the clause product with Φ_e replaced by $\partial_v \Phi_e$.

$$\sum_{\mathbf{x}} W_{e,v}(\mathbf{x}) \underbrace{\left(\sum_i b_i x_i \right)}_{\partial_T \ell_{e,v}(t)} = \sum_i b_i \underbrace{\sum_{\mathbf{x}: x_i=1} W_{e,v}(\mathbf{x})}_{\text{Part III with one bit fixed}}.$$

$J \leq o(n)$ ensures that derivative computation remains subexponential complexity.

The subexponential running time

An example way to set parameters is:

$$s \asymp \frac{(\log n)^{1/6}}{(\log \log n)^2}, \quad r \asymp \frac{\sqrt{\log n}}{(\log \log n)^2}, \quad Q \asymp \frac{\log n}{\log \log n}, \quad R \asymp \sqrt{\log n}.$$

- After sparsification, $m \leq Bn$ with $B = O((s \log s)^2) = o(r)$.
- The highest derivative order is $J = \lfloor 3mR/Q \rfloor = o(n)$.
- $n/2^{R^2/Q}$ outlier eigenvalues.

Final runtime:

$$\underbrace{2^{n/s}}_{\text{sparse formulas}} \cdot \underbrace{2^{n/2^{R^2/Q}}}_{\text{measurement guesses}} \cdot \underbrace{2^{2n/r}}_{\text{assignments per color}} \cdot \underbrace{\binom{n}{J} 2^J}_{\text{derivative restrictions}} = 2^{o(n)}.$$